Machine Learning Cloud Platforms

There are a number of machine learning cloud platforms, we provide *more* details about *a few* **below**. In the *next* few lessons, you will learn how to use **Amazon's SageMaker** to *deploy* machine learning models. Therefore, we *focused* on providing *more* information on [Amazon's SageMaker](https://aws.amazon.com/sagemaker/). To allow for a comparison of features offered by [SageMaker](https://aws.amazon.com/sagemaker/), we also provide detailed information about [Google's ML Engine](https://cloud.google.com/ml-engine/) because it's *most* similar to **SageMaker**.

## **Amazon Web Services (AWS)**

[Amazon Web Services (AWS) SageMaker](https://aws.amazon.com/sagemaker/) is Amazon's cloud service that allows you to *build*, *train*, and *deploy* machine learning models. Some advantages to using Amazon's SageMaker service are the following:

* **Flexibility in Machine Learning Software**: [SageMaker](https://aws.amazon.com/sagemaker/) has the flexibility to enable the use of ***any*** programming language or software framework for building, training, and deploying machine learning models in **AWS**. For the details see the three methods of modeling within **SageMaker** **below**.
  + [Built-in Algorithms](https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html) - There are at least fifteen built-in algorithms that are easily used within SageMaker. Specifically, built-in algorithms for discrete classification or quantitative analysis using [linear learner](https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner.html) or [XGBoost](https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html), item recommendations using [factorization machine](https://docs.aws.amazon.com/sagemaker/latest/dg/fact-machines.html), grouping based upon attributes using [K-Means](https://docs.aws.amazon.com/sagemaker/latest/dg/k-means.html), an algorithm for [image classification](https://docs.aws.amazon.com/sagemaker/latest/dg/image-classification.html), and many other algorithms.
  + **Custom Algorithms** - There are different programming languages and software frameworks that can be used to develop custom algorithms which include: [PyTorch](https://docs.aws.amazon.com/sagemaker/latest/dg/pytorch.html), [TensorFlow](https://docs.aws.amazon.com/sagemaker/latest/dg/tf.html), [Apache MXNet](https://docs.aws.amazon.com/sagemaker/latest/dg/mxnet.html), [Apache Spark](https://docs.aws.amazon.com/sagemaker/latest/dg/apache-spark.html), and [Chainer](https://docs.aws.amazon.com/sagemaker/latest/dg/chainer.html).
  + [Your Own Algorithms](https://docs.aws.amazon.com/sagemaker/latest/dg/your-algorithms.html) - Regardless of the programming language or software framework, you can use your own algorithm when it **isn't** included within the *built-in* or *custom algorithms* **above**.
* **Ability to Explore and Process Data within SageMaker**: [SageMaker](https://aws.amazon.com/sagemaker/) enables the use of [Jupyter Notebooks](https://docs.aws.amazon.com/sagemaker/latest/dg/nbi.html) to explore and process data, along with creation, training, validation, testing, and deployment of machine learning models. This notebook interface makes data exploration and documentation easier.
* **Flexibility in Modeling and Deployment**: [SageMaker](https://aws.amazon.com/sagemaker/) provides a number of features and automated tools that make ***modeling*** and ***deployment*** easier. For the details on these features within **SageMaker** see **below**.
  + [Automatic Model Tuning](https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning.html): [SageMaker](https://aws.amazon.com/sagemaker/) provides a feature that allows hyperparameter tuning to find the **best** version of the model for *built-in* and *custom algorithms*. For built-in algorithms **SageMaker** also provides evaluation metrics to evaluate the performance of your models.
  + [Monitoring Models](https://docs.aws.amazon.com/sagemaker/latest/dg/monitoring-overview.html): [SageMaker](https://aws.amazon.com/sagemaker/) provides features that allow you to monitor your *deployed* models. Additionally with *model deployment*, one can choose *how much* traffic to route to *each* deployed model (model variant). More information on routing traffic to model variants can be found [here](https://docs.aws.amazon.com/sagemaker/latest/dg/API_ProductionVariant.html) and [here](https://docs.aws.amazon.com/sagemaker/latest/dg/API_CreateEndpointConfig.html) .
  + **Type of Predictions**: [SageMaker](https://aws.amazon.com/sagemaker/) by *default* allows for [On-demand](https://docs.aws.amazon.com/sagemaker/latest/dg/ex1-test-model.html) type of predictions where *each* prediction *request* can contain *one* to *many* requests. **SageMaker** also allows for [Batch](https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-batch.html) predictions, and request *data* size limits are based upon S3 object size limits.

## **Google Cloud Platform (GCP)**

[Google Cloud Platform (GCP) ML Engine](https://cloud.google.com/ml-engine/) is Google's cloud service that allows you to *build*, *train*, and *deploy* machine learning models. Below we have highlighted some of the ***similarities*** and ***differences*** between these two cloud service platforms.

* **Prediction Costs**: The ***primary difference*** between the two is how they handle predictions. With ***SageMaker*** *predictions*, you must leave resources running to provide predictions. This enables *less* latency in providing predictions at the *cost* of paying for running *idle* services, if there are no (or few) prediction requests made while services are running. With ***ML Engine*** *predictions*, one has the option to *not* leave resources running which reduces cost associated with *infrequent* or *periodic* requests. Using this has *more* latency associated with predictions because the resources are in a offline state until they receive a prediction request. The *increased* latency is associated to bringing resources back online, but one *only* pays for the time the resources are *in use*. To see more about [ML Engine pricing](https://cloud.google.com/ml-engine/docs/pricing#node-hour) and [SageMaker pricing](https://cloud.google.com/ml-engine/docs/pricing#node-hour).
* **Ability to Explore and Process Data**: Another ***difference*** between **ML Engine** and **SageMaker** is the fact that *Jupyter Notebooks* are not available within **ML Engine**. To use *Jupyter Notebooks* within **Google's Cloud Platform** (GCP), one would use [Datalab](https://cloud.google.com/datalab/docs/). **GCP** separates data exploration, processing, and transformation into other services. Specifically, [Google's Datalab](https://cloud.google.com/datalab/docs/) can be used for data exploration and data processing, [Dataprep](https://cloud.google.com/dataprep/docs/) can be used to explore and transform raw data into clean data for analysis and processing, and [DataFlow](https://cloud.google.com/dataflow/docs/) can be used to deploy batch and streaming data processing pipelines. Noting that **Amazon Web Services** (AWS), also have data processing and transformation pipeline services like [AWS Glue](https://aws.amazon.com/glue/) and [AWS Data Pipeline](https://aws.amazon.com/datapipeline/).
* **Machine Learning Software**: The final ***difference*** is that [Google's ML Engine](https://cloud.google.com/ml-engine/) has *less* flexibility in available software frameworks for building, training, and deploying machine learning models in **GCP** as compared to **Amazon's SageMaker**. For the details regarding the two available software frameworks for modeling within **ML Engine** see **below**.
  + [Google's TensorFlow](https://cloud.google.com/ml-engine/docs/tensorflow/) is an open source machine learning framework that was originally developed by the Google Brain team. [TensorFlow](https://www.tensorflow.org/) can be used for creating, training, and deploying machine learning and deep learning models. [Keras](https://keras.io/) is a higher level API written in Python that runs on top of [TensorFlow](https://www.tensorflow.org/), that's easier to use and allows for faster development. GCP provides both [TensorFlow examples](https://cloud.google.com/ml-engine/docs/tensorflow/samples) and a [Keras example](https://cloud.google.com/ml-engine/docs/tensorflow/samples#census-keras).
  + [Google's Scikit-learn](https://cloud.google.com/ml-engine/docs/scikit/) is an open source machine learning framework in Python that was originally developed as a Google Summer of Code project. [Scikit-learn](https://cloud.google.com/ml-engine/docs/scikit/) and an [XGBoost Python package](https://xgboost.readthedocs.io/en/latest/python/index.html) can be used together for creating, training, and deploying machine learning models. In the in [Google's example](https://cloud.google.com/ml-engine/docs/scikit/training-xgboost), [XGBoost](https://xgboost.readthedocs.io/en/latest/python/index.html) is used for modeling and [Scikit-learn](https://cloud.google.com/ml-engine/docs/scikit/) is used for processing the data.
* **Flexibility in Modeling and Deployment**: [Google's ML Engine](https://cloud.google.com/ml-engine/) provides a number of features and automated tools that make ***modeling*** and ***deployment*** easier, ***similar*** to the those provided by **Amazon's SageMaker**. For the details on these features within **ML Engine** see **below**.
  + [Automatic Model Tuning](https://cloud.google.com/ml-engine/docs/tensorflow/hyperparameter-tuning-overview): [Google's ML Engine](https://cloud.google.com/ml-engine/) provides a feature that enables hyperparameter tuning to find the **best** version of the model.
  + [Monitoring Models](https://cloud.google.com/ml-engine/docs/tensorflow/monitor-training): [Google's ML Engine](https://cloud.google.com/ml-engine/) provides features that allow you to monitor your models. Additionally [ML Engine](https://cloud.google.com/ml-engine/) provides methods that enable [managing runtime versions](https://cloud.google.com/ml-engine/docs/tensorflow/versioning) and [managing models and jobs](https://cloud.google.com/ml-engine/docs/tensorflow/managing-models-jobs).
  + **Type of Predictions**: [ML Engine](https://cloud.google.com/ml-engine/) allows for [Online](https://cloud.google.com/ml-engine/docs/tensorflow/online-predict)(*or On-demand*) type of predictions where *each* prediction *request* can contain *one* to *many* requests. **ML Engine** also allows for [Batch](https://cloud.google.com/ml-engine/docs/tensorflow/batch-predict) predictions. More information about **ML Engine's** [Online and Batch predictions](https://cloud.google.com/ml-engine/docs/tensorflow/online-vs-batch-prediction).

## **Microsoft Azure**

Similar to [Amazon's SageMaker](https://aws.amazon.com/sagemaker/) and [Google's ML Engine](https://cloud.google.com/ml-engine/), Microsoft offers [Azure AI](https://azure.microsoft.com/en-us/overview/ai-platform/#platform). **Azure AI** offers an open and comprehensive platform that includes AI software frameworks like: [TensorFlow](https://www.tensorflow.org/), [PyTorch](https://pytorch.org/), [scikit-learn](http://scikit-learn.org/stable/), [MxNet](https://mxnet.incubator.apache.org/), [Chainer](https://chainer.org/), [Caffe2](https://caffe2.ai/), and other software like their [Azure Machine Learning Studio](https://azure.microsoft.com/en-us/services/machine-learning-studio/). For more details see [Azure AI](https://azure.microsoft.com/en-us/overview/ai-platform/#platform) and [Azure Machine Learning Studio](https://azure.microsoft.com/en-us/services/machine-learning-studio/).

## **Paperspace**

[Paperspace](https://www.paperspace.com/ml) simply provides GPU-backed virtual machines with industry standard software tools and frameworks like: [TensorFlow](https://www.tensorflow.org/), [Keras](https://keras.io/), [Caffe](http://caffe.berkeleyvision.org/), and [Torch](http://torch.ch/) for machine learning, deep learning, and data science. **Paperspace** claims to provide more powerful and less expensive virtual machines than are offered by **AWS**, **GCP**, or **Azure**.

## **Cloud Foundry**

[Cloud Foundry](https://www.cloudfoundry.org/) is an open source cloud application platform that's backed by companies like: Cisco, Google, IBM, Microsoft, SAP, and more. [Cloud Foundry](https://www.cloudfoundry.org/) provides a faster and easier way to build, test, deploy, and scale applications by providing a choice of clouds, developer frameworks, and applications services to it's users. [Cloud Foundry Certified Platforms](https://www.cloudfoundry.org/certified-platforms/) provide a way for an organization to have their cloud applications portable across platforms including [IBM](https://www.ibm.com/cloud/cloud-foundry) and [SAP](https://cloudplatform.sap.com/index.html) cloud platforms.